



Università degli Studi di Parma

Dipartimento di Ingegneria e Architettura

Sistemi operativi e in tempo reale - a.a. 2022/23

Scheduling clock-driven

prof. Stefano Caselli

stefano.caselli@unipr.it

Modello dei task



- L'approccio clock-driven è applicabile quando il sistema è *prevalentemente basato su task periodici* e deterministico
- Al più è possibile gestire *alcuni* task aperiodici o sporadici, che vengono integrati nel *framework deterministico*
- Modello dei task:
 - n task periodici, con n fisso
 - i *parametri* di tutti i task periodici sono *noti a priori*; variazioni nei tempi di inter-rilascio dei task periodici sono trascurabili
 - Ogni job $J_{i,k}$ di τ_i è *pronto* per l'esecuzione all'istante di rilascio $r_{i,k}$
 - \rightarrow Non ci sono ulteriori vincoli all'esecuzione dopo il rilascio del job

Task periodico



- Un *task periodico* τ_i è definito da una tupla del tipo:
$$\tau_i = (C_i, D_i, T_i)$$
- Ovvero da:
$$\tau_i = (C_i, D_i, T_i, \Phi_i)$$
- Spesso:
$$\Phi_i = 0 \quad \text{e} \quad D_i = T_i$$
- Nei casi più semplici e frequenti, un task è caratterizzato solo da tempo di esecuzione e periodo: $\tau_i = (C_i, T_i)$ o $\tau_i = (C_i, p_i)$
- *Fattore di utilizzazione*: $U_i = C_i/T_i$

Scheduler statico timer-driven



- ❑ La predisposizione off-line di *schedule statiche* permette l'adozione durante la progettazione di *algoritmi sofisticati*, anche complessi
- ❑ I periodi in cui il processore è inattivo possono essere resi disponibili per eventuali job aperiodici
- ❑ \Rightarrow Nelle ipotesi date, è *possibile* adottare anche una schedulazione *non work-conserving*
- ❑ Quali i potenziali vantaggi/motivazioni?

Scheduler statico timer-driven



- Prima fase: predisposizione off-line di una *schedule statica*, anche con algoritmi sofisticati e di elevata complessità
- E' possibile tenere conto di criteri di priorità specifici, adattando la *schedule* di conseguenza
- La realizzazione può essere basata su una **tabella** (*scheduling table-driven*) costituita dalle coppie $(t_k, J(t_k))$, ove t_k è l'istante di decisione e $J(t_k)$ è il job da mettere in esecuzione all'istante t_k (oppure ϕ , idle)
- Essendo i task periodici, la tabella avrà un *numero finito di entry*

Scheduler clock-driven



- Input per lo schedulatore: Schedule $(t_k, J(t_k))$, $k=0,1,\dots,N-1$ (N =num. attivazioni *job* e intervalli idle), iperperiodo H

Task Scheduler:

$i:=0$; $k:=0$;

<imposta il timer all'istante t_0 >

do forever:

<attendi il timer interrupt>

$i:=i+1$; // prepara indici e timer per prossimo evento

$k:=i \bmod N$;

<imposta il timer all'istante $\lfloor i/N \rfloor H + t_k$ >

if $J(t_{k-1})$ è vuoto // esecuzione evento attuale $k-1$

then wakeup(aperiodic)

else wakeup($J(t_{k-1})$) // job in tabella, entry $k-1$

end

end Scheduler;

Scheduler clock-driven



□ Notazione

- $\lfloor i/N \rfloor$ = div risultato intero della divisione (es: $21 \text{ div } 10 = 2$)
- $i \text{ div } N$ = numero di iperperiodi già completati
- t_k tempo relativo all'interno dell'iperperiodo corrente, ottenuto accedendo alla entry k della tabella prememorizzata

□ Caratteristiche:

- Questa struttura di executive consente di gestire schedule non suddivise in frame periodici
- Ogni t_i è l'inizio di un job o di una fase programmata di idle (in cui saranno eseguiti task aperiodici o in background)

Scheduler clock-driven



□ Quali problemi presenta questo approccio?

```
Task Scheduler: //input:  $Schedule(t_k, J(t_k)), k=0, \dots, N-1; H$   
i:=0; k:=0;  
<imposta il timer all'istante  $t_0$ >  
do forever:  
  <attendi il timer interrupt>  
  i:=i+1; // prepara indici e timer per prossimo evento  
  k:=i mod N;  
  <imposta il timer all'istante  $\lfloor i/N \rfloor H + t_k$ >  
  if  $J(t_{k-1})$  è vuoto // esecuzione evento attuale k-1  
  then wakeup(aperiodic)  
  else wakeup( $J(t_{k-1})$ ) // job in tabella, entry k-1  
end  
end Scheduler;
```


Scheduler clock-driven



- ❑ Problemi:
 - Intervalli molto brevi? (es. di idle)
 - Necessità di reimpostare il timer ogni volta, possibile accumulo errori
 - Un solo job per volta, può risultare tabella «lunga»!
 - Controlli di correttezza non integrati: Nuovo job rilasciato? Completato il precedente?
 - Manca politica di gestione errori (overrun e mancati rilasci): Abort? Extend time? Count error? Notify?

- ❑ Nel seguito: schema generale di executive per scheduling clock-driven basato su *frame periodici*

Executive ciclico



- ❑ I *task periodici* sono una parte importante, spesso prevalente, di molte applicazioni real-time
- ❑ Un *executive ciclico* è un programma di controllo che realizza in modo esplicito la *sequenza di esecuzione* (interleaving) di *task periodici* su una singola CPU
- ❑ L'interleaving viene realizzato in modo deterministico, in modo che i tempi di esecuzione siano garantiti
- ❑ L'interleaving dei job è definito in base ad una *schedule ciclica*



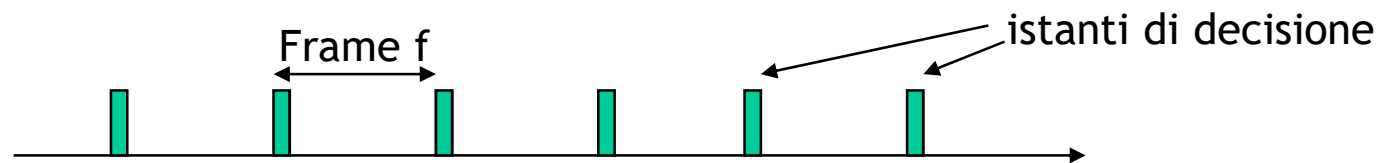
Executive ciclico

- Proprietà importanti dell'executive ciclico:
- Assenza di preemption
 - Non richiede di eseguire preemption a run-time
 - Le commutazioni sono realizzate in istanti pre-pianificati, e quindi “sicuri”
- Complessità offline
 - È possibile utilizzare algoritmi complessi, che comunque vengono eseguiti offline, per generare una schedule che soddisfi gli obiettivi di performance e di correttezza
- Rigidità
 - Small change? Do it again
 - Modifiche sostanziali possibili solo offline

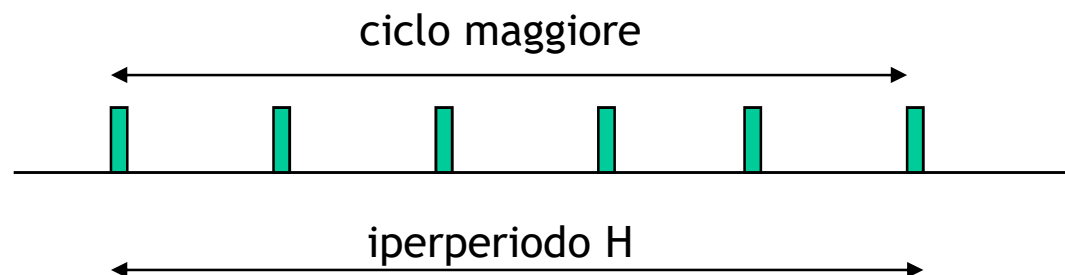
Struttura generale



- Istanti di decisione *periodici*:



- Le decisioni di scheduling sono prese periodicamente:
 - scelta del job o dei job da eseguire
 - operazioni di *monitoring* e di *garanzia*
- *Ciclo maggiore*: sequenza di *frame* nell'iperperiodo





Il ciclo maggiore

- Una *schedule ciclica* comprende una o più *schedule principali*, che descrivono la sequenza di azioni da eseguire durante un intervallo di tempo prefissato (*ciclo maggiore*)
- Le azioni di una *schedule principale* sono eseguite periodicamente
- La durata del ciclo maggiore è pari al minimo comune multiplo dei periodi dei task che fanno parte di ciascuna delle *schedule principali* (*iperperiodo*, H)
- *Schedule principali* diverse corrispondono a *modi* di funzionamento diversi del sistema, tra cui si commuta in corrispondenza a specifici eventi real-time

Frame



- ❑ Ciascuna schedule principale è divisa in una o più *schedule secondarie* o *frame*
 - I limiti temporali di un frame corrispondono ad istanti in cui si verifica un interrupt hardware generato da un timer e in cui vengono imposti e verificati i *vincoli temporali*
- ❑ A ciascun frame è allocato un intervallo di tempo fisso durante il quale deve eseguire una *sequenza di job* (*scheduling block*)
- ❑ Se i job di un frame sono completati in anticipo, il processore attende inattivo o esegue job in background fino all'inizio del successivo frame
- ❑ Se i job di un frame non sono completati in tempo, il sistema rileva un errore di *frame overrun*



Funzioni del frame

- ❑ gestione del temporizzatore (eventuale)
 - ❑ dispatching dei job
 - ❑ abilitazione all'esecuzione di job in background
 - ❑ verifica del rispetto delle deadline e dei rilasci dei job
 - ❑ gestione di errori ed eccezioni
-
- ❑ non c'è preemption all'interno di un frame!

Il ciclo minore



- In un executive ciclico i *frame* sono *tutti di uguale durata (ciclo minore)*: la verifica dei vincoli temporali del frame è realizzata mediante la gestione degli eventi generati da un *timer periodico preimpostato*
- Requisito: per la verifica dei vincoli temporali, la *durata massima di ciascun frame* non può essere superiore al periodo minimo (più in generale, alla deadline relativa minima) dei job da eseguire entro il frame

Il ciclo minore



- In un executive ciclico i *frame* sono tutti di uguale durata (*ciclo minore*): la verifica dei vincoli temporali del frame è realizzata mediante la gestione degli eventi generati da un *timer periodico preimpostato*
- Requisito: per la verifica dei vincoli temporali, la *durata massima di ciascun frame* non può essere superiore al periodo minimo (più in generale, alla deadline relativa minima) dei job da eseguire entro il frame
- A causa del requisito sulla dimensione massima del frame, eventuali job con tempo di esecuzione superiore devono essere suddivisi in *sotto-job*, ciascuno dei quali di durata tale da completare entro un frame --> *job partitioning*



Durata del ciclo minore: requisiti

- m = *durata del ciclo minore o dimensione del frame*,
 M = *durata del ciclo maggiore, o iperperiodo*

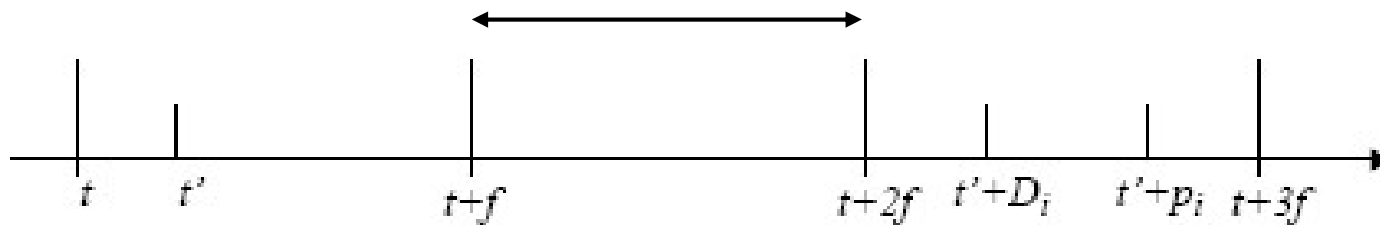
- Devono essere soddisfatti simultaneamente i seguenti requisiti:
 - (1) $m \leq D_i \quad \forall i$
 - (2) $m \geq c_i \quad \forall i$
 - (3) $M/m = \lfloor M/m \rfloor$
 - (4) $m + (m - \text{MCD}(m, p_i)) \leq D_i \quad \forall i$

- Nota: Il vincolo (4) *sussume* (1)



Vincoli sulla dimensione del frame

- La dimensione del frame deve consentire di iniziare e completare ogni job all'interno di un medesimo frame
- m è un divisore intero di H (la condizione m è un divisore intero di almeno un periodo p_i non copre tutti i casi)
- Per monitorare il *rispetto dei vincoli temporali*, i frame devono essere sufficientemente brevi, in modo tale che tra l'istante di rilascio e la deadline di ciascun job si verifichi almeno un frame completo:



Dimensione del frame

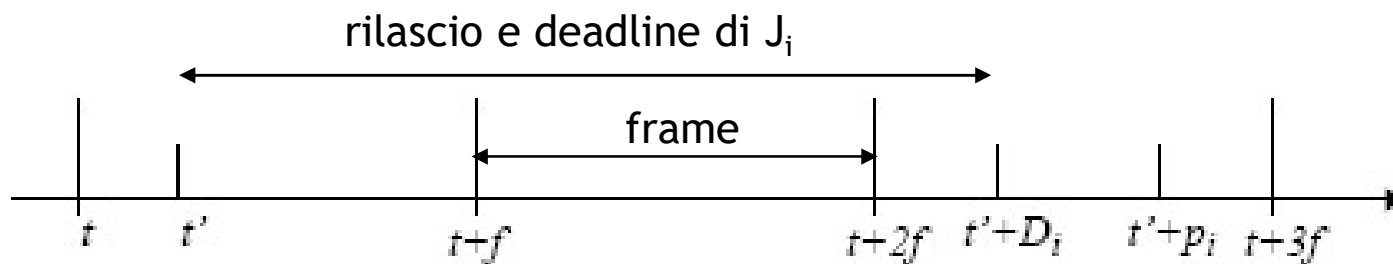


- Queste scelte:
 - evitano la preemption
 - minimizzano la lunghezza totale della schedule
 - assicurano che ogni iperperiodo ospiti un numero intero di frame
 - consentono la verifica del rispetto delle deadline e dei rilasci dei job (ovvero dei vincoli temporali) *all'inizio di ciascun frame*
 - non impongono relazioni di fase specifiche tra i task, che sono spesso impossibili da realizzare



Vincoli sulla dimensione del frame

- Vincolo: $m + (m - \text{MCD}(m, p_i)) \leq D_i \quad \forall i$
- L'esecuzione c_i di ciascun job J_i deve essere garantita da un frame (\rightarrow di durata almeno pari al $\max c_i$):
 - il job deve essere attivato dopo l'inizio di un frame
 - il termine del medesimo frame ne deve verificare il completamento prima della deadline
 - ovvero: $m + (m - \min(t' - t)) \leq D_i$





Caso peggiore per l'istante di rilascio

- ❑ Il caso peggiore è quello a scostamento minimo tra l'istante di rilascio t' e l'inizio t del frame precedente
 - ❑ Una proprietà della *teoria dei numeri*:
Date due sequenze periodiche, con periodi X e Y , la distanza minima tra due valori diversi delle sequenze è data dal *massimo comune divisore* dei periodi, $MCD(X,Y)$
 - Identifica il caso peggiore dal punto di vista del contenimento di una intera esecuzione entro il frame
 - Esclude il caso in cui i valori sono uguali, che sarebbe più favorevole
 - ❑ Riportiamo il problema ad un problema sugli interi
-



Esempio

- Calcolo della dimensione del frame per il seguente set di task
 $T_i = (p_i, c_i, D_i)$:
 $T1 = (15, 1, 14)$ $(D_i \neq p_i)$
 $T2 = (20, 2, 26)$
 $T3 = (22, 3, 22)$
- Iperperiodo: $H = 660$
- Vincoli:
 - (1) $m \leq D_i \quad \forall i \quad \rightarrow m \leq 14$
 - (2) $m \geq c_i \quad \forall i \quad \rightarrow m \geq 3$
 - (3) m divide $H \quad \rightarrow m = 2, 3, 4, 5, 6, 10, 11, 12, \dots$
 - (4) $2m - \text{MCD}(m, p_i) \leq D_i \quad \forall i \quad \rightarrow m = 2, 3, 4, 5, 6$
- ➔ Possibili valori per m : 3, 4, 5, 6

Suddivisione dei job



- Talvolta i parametri non consentono di rispettare tutti i vincoli simultaneamente
- E' possibile rilassare il vincolo (2) suddividendo i job in sotto-job (*job partitioning*)
- Esempio [$T_i=(p_i, c_i, D_i)$]:
 - T1=(4, 1, 4) (2) $\Rightarrow m \geq 5$
 - T2=(5, 2, 5) (1) $\Rightarrow m \leq 4$???
 - T3=(20, 5, 20)
- Dove operare il partizionamento? In generale, sarà dipendente dalla applicazione!

Suddivisione dei job



- Se è possibile suddividere T3 in tre job elementari con tempi di esecuzione 1+3+1:

T1 =(4, 1, 4)

T2 =(5, 2, 5)

T31=(20, 1, 20)

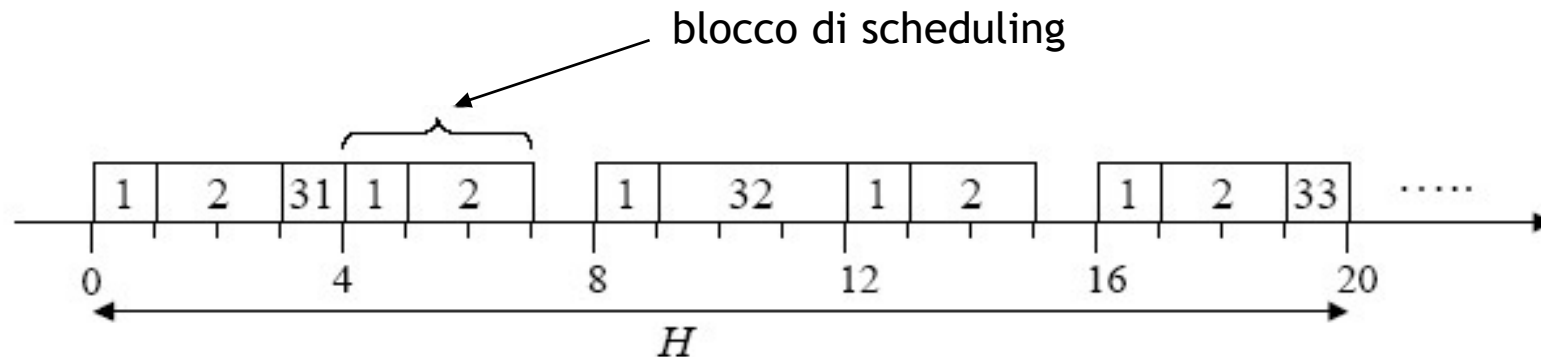
T32=(20, 3, 20)

T33=(20, 1, 20)

(1) $\Rightarrow m \leq 4$

(2) $\Rightarrow m \geq 3$

Ad es., $m=4$



Suddivisione dei job



- Se è possibile suddividere T3 in tre job elementari con tempi di esecuzione 1+3+1:

T1 =(4, 1, 4)

T2 =(5, 2, 5)

T31=(20, 1, 20)

T32=(20, 3, 20)

T33=(20, 1, 20)

(1) $\Rightarrow m \leq 4$

(2) $\Rightarrow m \geq 3$

Ad es., $m=4$

Istanti di decisione a inizio frame:
Job frame precedente completati?
Job nuovo frame rilasciati?

